REMARKS/ARGUMENTS

Amendments were made to the specification to correct errors and to clarify the specification. No new matter has been added by any of the amendments to the specification.

Claims 1-9 are pending in the present application. Claims 1, 4, 7 and 8 have been amended, and Claim 9 has been added, herewith. Reconsideration of the claims is respectfully requested.

I. Objection to Specification

The disclosure was objected to as containing embedded hyperlinks. Applicants have amended the Specification to remove such objected-to hyperlinks.

The disclosure was objected to at page 2, lines 20-30 as containing incomplete information. Applicants have amended the Specification to include such information.

The title of the invention was objected to as not being descriptive. Applicants have amended the title in accordance with the Examiner's suggestion.

Therefore, the objection to the Specification has been overcome.

II. 35 U.S.C. § 101

Claims 1-8 stand rejected under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

In rejecting Claims 1-8 under 35 U.S.C. § 101, the Examiner states such claims are nonstatutory as they disclose a claimed invention that is an abstract idea as defined in the case *In re Warmerdam*, 33 F 3d 1354, 31 USPQ 2d 1754 (Fed. Cir. 1994). Applicants have amended Claim 1 to re-emphasize that the claimed steps are computer-implemented steps (i.e. a program is being executed by the computer), and thus Claim 1 (and dependent Claims 2 and 3) is not a mere abstract idea.

With respect to Claim 4 (and dependent Claims 5 and 6), such claim specifically recites an apparatus for executing a program, and such an apparatus is a machine – which is one of the expressly enumerated categories of statutory subject matter per 35 U.S.C. 101[1].

With respect to Claim 7, and per MPEP 2106(IV)(B)(1)(a), "a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's

[1] 35 U.S.C. 101: Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

functionality to be realized, and is thus statutory". It is urged that Claim 7 complies with this MPEP requirement.

With respect to Claim 8, the Examiner characterizes such claim at the top of page 3 of the present Office Action as being a "method for executing a program". Applicants urge that, to the contrary, Claim 8 is instead directed to "A compiler", and the computer-readable program code is a concrete, useful and tangible result that is generated by such compiler. Thus, contrary to the Examiner's rejection of Claim 8, such claim is not directed to a "method for executing a program", and accordingly such claim has been erroneously rejected under 35 U.S.C. § 101 due to such erroneous characterization of Claim 8.

Therefore, the rejection of Claims 1-8 under 35 U.S.C. § 101 has been overcome.

III. 35 U.S.C. § 112, Second Paragraph

Claims 1-8 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter, which applicants regard as the invention. This rejection is respectfully traversed.

In rejecting Claims 1-8, the Examiner states that these claims are incomplete for omitting essential steps of linking, loading, returning control, executing, etc. Applicants urge that Claim 1 is directed to a method of 'executing' a computer program, and any 'linking' or 'loading' pre-processing steps that may occur in the creation of such program is not an essential step in the claimed invention - which is instead directed to the actual program execution of such program. Restated – how the program is actually built or formed is not what the invention of Claim 1 is directed to, which instead is directed to the actual execution of a program however it may have been built/formed. Thus, any linking and/or loading steps are not essential steps of Claim 1. Applicants have amended Claim 1 to explicitly recite the control and processing steps.

Applicants traverse the rejection of Claims 2-8 for similar reasons to those given above with respect to Claim 1.

Therefore, the rejection of Claims 1-8 under 35 U.S.C. § 112, second paragraph has been overcome.

IV. 35 U.S.C. § 102, Anticipation

Claims 1-8 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Agesen et al. US Patent No. 7,117,481. This rejection is respectfully traversed.

Generally, the claims are directed to techniques for improving overall system throughput through use of a novel approach for *speculatively executing computer code* (Specification page 11, lines 9-17). As is typical in many types of computer programs, when a function is called, the function is executed and

when the function execution is completed, the function returns a return code to the calling program that called the function. The calling program then typically invokes or executes a particular subsequent routine, chosen from a plurality of possible subsequent routines, based on what the return code value is. For example, a return code of zero (0) may indicate normal processing was performed by the called routine, whereas a return code of minus-one (-1) may indicate that the called function encountered some type of error condition. The calling program, which called the called function, would typically execute or invoke different code depending upon whether a 0 of a -1 was returned as a return value by the called function. For example, the calling program may invoke the display or logging of an error message if the called function returned a return value of minus-one (-1), and then terminate due to such error. On the other hand, the calling program may continue with normal processing if the called function returned a return value of zero (0), which indicates a successful completion of the called function. Per the features of Claim 1, both (1) the subsequent error condition processing (which is code associated with a minus-one (-1) return value) and (2) the normal processing (which is code associated with a zero (0) return value) are speculatively invoked *even though the called function is still being processed and hence has not yet returned a return value.* Then, once the called function completes processing and returns the actual return value, the speculatively executed code which corresponds to this actual value, which is returned at the completion of the called function, is chosen for continued processing. Since this speculative code has already executed to some extent during the same time that the called function was being processed, even though the return value being returned by the called function was not yet known, an overall improvement in system performance is obtained since some or all of the speculatively executed code has already been executed without having to wait on the actual results (return code) returned by the called function.

For example, Claim 1 recites "processing, on a first thread, a function defined by the function call, the function having one or more programmer predefined typical return values". As can be seen, this function is the 'called function' described above that is processed and which returns a return value. Claim 1 further recites "for each predefined return value, pre-processing on an additional thread, the one or more subsequent instructions". As can be seen, for *each of the possible return values* that might be returned by the first thread, an additional thread is pre-processed such that, advantageously, the subsequent processing associated with a given return value can be processed before the first thread actually returns the return value – thus speculatively executing this instruction(s) that would otherwise be conditionally processed depending upon the return value returned by the first thread (the 'called function'), thereby advantageously providing for *parallel processing of (i) the code providing the return code, as well as (ii) the subsequent return-code-specific code, notwithstanding that the actual return code has not yet been returned by the function providing the return code.* Such parallel processing enhances overall system performance as the subsequent return-code-specific code does not have to wait until the first thread

completes processing and returns an actual return value, but instead such code is speculatively executed while the first thread (called function) continues processing.

In rejecting the claimed feature of "for each predefined return value, pre-processing on an additional thread, the one or more subsequent instructions", the Examiner states that Agesen teaches such feature at col. 18, lines 51-62 and col. 15, lines 56-60. Agesen states at col. 18, lines 51-62:

> "By working through the definitions of the functions Lock, Unlock, sema_Wait and sema_Signal, and noting the different states of the lock (S,nthreads), one can establish the correctness of the composite lock according to the invention even in the case of interleaved contention involving more than two concurrent threads. One can also satisfy oneself that this is true by considering the following: Assume multi-thread interleaving. When the thread that currently holds the lock calls Unlock, then either at least one other thread is currently suspended, or none is. Each suspended thread will be awakened in turn by a previously awakened thread, until no threads remain suspended."

and Agesen states at col. 15, lines 56-60:

> "Atomic_incr(k): A preferably (but not necessarily) hardware-provided atomic function that increments the value of k but that returns the value of k immediately before it is incremented. The preferred implementation of this atomic operation is thus "fetch-and-increment." (Modifications to the front-end lock component code to accommodate returning the post-incrementing value of k will be obvious to skilled programmers.)"

Neither of these passages describe any code processing that is associated with a return value for the first thread (such first thread alleged to be taught by Agesen's first thread at col. 17, lines 35-39). Importantly, this Agesen first thread does not describe any type of return value at all, and thus it necessarily follows that Agesen does not teach 'for each predefined return value' (of the first thread), pre-processing on an additional thread ....' as there is no return value(s) described for the Agesen first thread. The return value of k that is described at Agesen col. 15 is not a return value for the alleged first thread (which is stated by the Examiner to be the first thread described at Agesen col. 17, lines 35-39), but rather is a return value of an Atomic_incr function. Even assuming Atomic_incr(k) was construed to be the claimed first thread, the cited reference does not teach pre-processing on an additional thread any code associated with this 'k' return value. Applicants urge that the amendment to Claim 1 further emphasizes these thread/return code distinctions. Quite simply, the cited reference *does not teach <u>any</u> type of speculative code execution,*

either as per the particular implementation as recited in Claim 1, or any other type of implementation of speculative code execution. Thus, as every element recited in Claim 1 is not identically shown in a single reference, it is urged that Claim 1 is not anticipated by the cited reference.

Applicants traverse the rejection of Claims 2 and 3 for reasons given above with respect to Claim 1 (of which Claims 2 and 3 depend upon).

Applicants traverse the rejection of Claims 4-8 for similar reasons to those given above with respect to Claim 1.

Therefore, the rejection of Claims 1-8 under 35 U.S.C. § 102(e) has been overcome.

V. New Claim

Claim 9 has been added herewith, and Examination of such claim is respectfully requested.

VI. Conclusion

It is respectfully urged that the subject application is patentable over the cited reference and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: May 18, 2007

Respectfully submitted,

/Wayne P. Bailey/

Wayne P. Bailey
Reg. No. 34,289
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants